

**REMARKS**

All pending claims, claims 22-51, stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent 5,193,1880 (Hastings) in view of U.S. Patent 5,822,590 (Gupta). The Applicants respectfully traverse for the reasons set forth below.

**1. Prior art must suggest the desirability of the claimed invention.**

Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See MPEP 2143.01.

With regard to the motivation to combine references, the Examiner states on page 4, lines 2-9 of the outstanding office action,

It would have been obvious . . . to provide the load instructions as taught by Gupta's persistent model language to the set of inserted instructions as suggested by Hastings to access and update persistently stored objects, and to load objects persistent objects [*sic*] into memory because memory accessed objects cannot limit to just program memory allocation and should be persisted in more non-volatile storage in order to alleviate burden on volatile memory, thus efficiently preserve resources that would be used for memory fault checking as intended by Hastings.

Thus, it appears that the Examiner is effectively saying that, to "efficiently preserve resources" and reduce burden on volatile memory so that it can be used for Hastings' memory fault checking, it would have been obvious to add Gupta's load instructions to the set of inserted instructions taught by Hastings. Applicants respectfully traverse.

Firstly, there is no suggestion in Gupta that load instructions (as imputed by the Examiner) of Gupta in any way increase available volatile memory so that they could be used for other purposes. The purpose of Gupta is rather to provide increased complex object heap space, beyond that available by *virtual memory* by extending the language syntax to include additional keyword and operator thereby adding increased functionality of an existing language without rewriting the compiler. See Abstract. Gupta does not mention "alleviate burden on volatile memory" nor is there any suggestion that Gupta's method is any less burdensome on virtual memory than the prior art approaches (see col. 1, lines 21-35). *Virtual memory* is a computer

design feature that permits software to use more memory than the computer physically possesses, typically using a swap file or swap partition in non-volatile memory.

Secondly, there is no suggestion in Hastings that the memory fault checking statements incorporated into the modified object code of a computer program place a significant burden on memory resources as to saturate available virtual memory, thereby requiring such techniques as taught by Gupta. Since there is no suggestion that the memory fault checking statements place a significant burden on system resources, it would not have been suggested to a person having ordinary skill in the art to take the step of indicating objects are persistible using the keywords provided by Gupta and performing source code translation in the manner proposed by Gupta.

Since the prior art and the general knowledge of a person having ordinary skill in the art lacked motivation to combine Hastings with Gupta in the manner set forth in the outstanding rejection, Applicants submit that claims 22-51 are allowable, and respectfully requests early allowance of same.

## **2. Reasonable Expectation of Success is Required.**

The prior art can be modified or combined to reject claims as prima facie obvious as long as there is a reasonable expectation of success. See MPEP 2143.02. In the present case, the step of compiling as taught by Hastings will fail because it will not recognize keywords introduced into the source code as required by Gupta so that the load instructions can be inserted. The only way to avoid this would be to modify the compiler which is specifically and purposefully avoided by Gupta.

Gupta teaches processing source code and only source code would work with the Gupta method since it relies on the introduction by the programmer/user of keywords that would not be recognized by the compiler. Gupta then processes this source code, which is termed, "the dbX program" into modified source code ("the X program") that is then ready to be compiled.

The Examiner suggests it would have been obvious "to provide the load instructions as taught by Gupta's persistent model language to the set of inserted instructions as suggested by Hastings. . . ." (Page 4, lines 3-4.) However, Hastings teaches that instructions are inserted into the *object* code (*after* compiling) while Gupta requires extra-syntactical keywords be utilized to generate inserted code (see column 5, lines 13-36). Thus, Hastings and Gupta represent two different approaches to two different problems which are wholly mutually exclusive. One simply

cannot insert extra-syntactical keywords into source code, and expect to successfully compile them. Nor can instructions inserted by Gupta be translated into object code and inserted there since the Gupta preprocessor relies on these extra-syntactical keywords.

If the Examiner were to suggest that the compiler be rewritten to accept the new keywords taught by Gupta, then there would be no need for preprocessing as taught by Gupta since compiler can interpret them and generate the proper object code without any pre- or post-processing. This would obviate the need for Gupta since the whole point of Gupta is to provide support for persistible complex object heap space without rewriting the compiler. See, e.g., the Abstract and column 2, line 66 to column 3, line 3.

Since the combination of Hastings and Gupta in the manner proposed in the outstanding office action would not have been successful in achieving the desired result, Applicants respectfully request that the rejection under 35 U.S.C. §103(a) be withdrawn, and that claims 22-51 be allowed.

### **3. All Claim Limitations Must Be Taught or Suggested**

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. See MPEP §2143.03. Therefore, to overcome a rejection based on obviousness under 35 U.S.C. §103(a), all that is required of Applicants is to point to any single limitation in the claim that does not have support in the prior art. Therefore, by way of example, Applicants respectfully point to the following elements present in claim 22, but not shown in the prior art. This list is obviously not necessarily exhaustive of all claim elements that distinguish the invention from the prior art, as any one of these elements by itself is sufficient to overcome the obviousness rejection set forth in the outstanding office action.

i) *“scanning the initial computer program to automatically identify object accessing instructions and corresponding program locations at which additional instructions are to be added”* (Claim 22, lines 7-9)

In contrast to the above-quoted element of claim 22, Hastings scans each module for memory access and inserts instructions for monitoring this access for debugging purposes. Hastings does not address data structures other than arrays and free variables. Objects are not mentioned. In fact, Hastings exclusively uses the word, “object” as an adjective in such phrases as “object code” or “object file.” The purpose of Hastings, i.e., to identify programming errors

that result in array bounds violations, uninitialized memory reads, free memory access, etc. (see col. 2, lines 25-50) would be inconsistent with data objects.

Likewise, Gupta does not scan to identify object accessing instructions. Gupta searches instead for keywords defined by the dbX language extensions of the disclosure. Specifically, Gupta searches for instances of “dbvar”, “dbnew()”, and pointer references (for triggering dbfaulting --- see col. 4, lines 42-47) .

Since Neither Hastings nor Gupta teach or suggest, “scanning . . . to automatically identify object accessing instructions” Applicants respectfully submit that claim 22 is allowable. Claims 23-25 depend from claim 22 and are allowable for the same reason as claim 22. Each remaining independent claim 26, 28, 32, 36, 38, 42, 46, and 48 contains similar limitations in lines 8-10, 4-6, 12-14, 13-15, 10-12, 14-16, 14-16, and 12-14, respectively. All line numbers are consistent with claims as provided in the attached Appendix. Thus, all pending claims should be allowed and early allowance of same is respectfully requested.

ii) *“automatically, under computer program control, revising the initial computer program . . . by modifying data structures of the persistent objects and adding object loading instructions”* (claim 22, lines 10-12)

Hastings is not related to objects and therefore has nothing to do with data structures of persistent objects. Gupta does not modify the data structures. The Examiner’s rationale is as follows:

“ . . . in view of the teachings by Gupta, some pointer structures are re-addressed in order to accommodate to changes related to persistent objects acknowledged in the preprocessing of the initial program ( e.g. *dbpointer is made to point at* – col. 4, lines 6-58; col. 6, lines 6-41); and some instructions are already allocated for accessing persistent objects when initially submitted for preprocessing. . . .”

(Page 4, lines 16-20.) Applicants respectfully disagree. Specifically, *dbpointer* is not a data structure of a persistent object as required by the above-quoted claim element. Rather, *dbpointer* “is a memory pointer whose lvalue is used to provide a hook to an OID [object identifier of a persistible object].” See col. 3, lines 28-47. Thus, *dbpointer* merely directs one to the address of an object identifier of the persistible object, and is not actually part of the persistible object itself. In contrast, the present disclosure, at page 14, lines 4-8, discloses, “The postprocessor 206 modifies the object data structures of the object classes that will be used to store main memory


copies of persistently stored objects to enable the storage of additional information required for managing object pointers and for keeping track of "dirty" objects."

Since Neither Hastings nor Gupta teach modifying data structures of the persistent objects, Applicants submit that that claim 22 is allowable over them. Claims 23-25, which depend from claim 22 are allowable for the same reason as claim 22. Furthermore, each remaining independent claim 26, 28, 32, 36, 38, 42, 46, and 48 contains limitations similar to the above quoted limitation at lines 11-13, 10-13, 15-17, 16-19, 13-16, 17-19, 17-19, and 15-18, respe. Therefore, all pending claims should be allowed and early allowance of same is respectfully requested.

In view of the remarks above, Applicants respectfully submit that the present application is in condition for allowance. A Notice of Allowance is therefore respectfully requested.

If the Examiner has any questions concerning the present amendment, the Examiner is kindly requested to contact the undersigned at (408) 749-6900 x6933. If any other fees are due in connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. ). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,  
MARTINE & PENILLA, LLP

  
Leonard E. Heyman, Esq.  
Reg. No. 40,418

710 Lakeway Drive, Suite 170  
Sunnyvale, CA 94085  
Telephone: (408) 749-6900  
Facsimile: (408) 749-6901